RLG-DHP System Master Specification v1.1

Luke Doel

March 2026

1 System Overview

The RLG-DHP architecture defines a recursive state-transition system composed of three coupled subsystems[cite: 58]:

- **Archetype Codex (AC):** The interpretation controller[cite: 5, 59].
- **RLG Engine (RLG):** Manages state-transition dynamics[cite: 6, 60, 62].
- **DHP Layer (DHP):** The decision and execution compiler[cite: 7, 63, 64].

The system forms a closed feedback loop over state space [cite: 65], where each cycle transforms the system state R_t into R_{t+1}[cite: 9]:

$$R_t \to AC \to RLG \to DHP \to R_{t+1} \tag{1}$$

2 Core Definitions

2.1 Archetype Codex (AC)

The AC is a mapping function $AC : R \to A_i$[cite: 119]. It defines the interpretation frame and modulates state perception[cite: 76, 120].

$$A_t = AC(R_t) \text{ [cite: 14, 75]} \tag{2}$$
$$R'_t = Interpret(R_t|A_t) \text{ [cite: 16, 79]} \tag{3}$$

This step defines relevance weighting and constraint activation[cite: 80].

2.2 RLG Engine

The RLG engine defines state transition dynamics and is blind to semantics, operating purely on structural transformation[cite: 125, 127].

$$R_L = Delay(R'_t) \quad (\text{where } L = lag) \text{ [cite: 19, 91, 94]} \tag{4}$$
$$G = \nabla C(R_L) \quad (\text{where } G = gradient) \text{ [cite: 21, 92, 96]} \tag{5}$$
$$B = Resolve(R_L, G) \quad (\text{where } B = behaviour) \text{ [cite: 23, 93, 98]} \tag{6}$$
$$R''_t = B \text{ [cite: 25, 100]} \tag{7}$$

2.3 DHP Layer

DHP is a constraint-based execution compiler that enforces governance rules and output validity[cite: 129, 131].

$$D_t = DHP(B, A_t, C) \text{ [cite: 28, 104]} \tag{8}$$
$$Execute(D_t) \to E \quad (\text{where } E = external\ state) \text{ [cite: 30, 109, 110]} \tag{9}$$
$$R_{t+1} = Observe(E) \text{ [cite: 32, 112, 118]} \tag{10}$$

3 System Invariants

- **Recursive Closure:** $R_{t+1} \rightarrow R_{t+k}$; all outputs feed forward as inputs[cite: 36, 135, 138].
- **Layer Separation:** $AC \neq RLG \neq DHP$; no layer may bypass another[cite: 37, 139, 143].
- **Mandatory Lag:** $L > 0$; all transitions include a delay component[cite: 38, 144, 146].
- **Gradient Necessity:** $G \neq 0 \Rightarrow B$ defined; behavior requires a constraint differential[cite: 39, 148, 150].
- **Execution Gating:** Only the DHP layer interacts with the external system[cite: 40].

4 Implementation Pseudocode

The following function defines the continuous execution of the architecture [cite: 41-52, 153-166]:

```
function RLG_DHP_SYSTEM(R):
    A = Archetype_Select(R)          // AC Layer
    R_int = Interpret(R, A)
    R_lag = Apply_Lag(R_int)         // RLG Engine (L)
    G = Compute_Gradient(R_lag)      // RLG Engine (G)
    B = Resolve_Behaviour(R_lag, G)  // RLG Engine (B)
    D = DHP_Compile(B, A, constraints) // DHP Layer
    Execute(D)
    R_next = Observe_World_State()
    return RLG_DHP_SYSTEM(R_next)    // Recursive Loop
```

5 Termination Condition

None. The system is inherently recursive[cite: 54].